МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи
«Робота з текстовою інформацією у .NET Framework»

з курсу «Кросплатформне програмування»
для студентів спеціальності 122 – Комп’ютерні науки

Затверджено редакційно-видавничою
радою університету,
протокол № 1 від 22.06.2017.

Харків
НТУ «ХПІ»
2017

Методичні вказівки до лабораторної роботи «Робота з текстовою інформацією у .NET Framework» з курсу «Кросплатформне програмування» для студентів спеціальності 122 – Комп’ютерні науки / Уклад. І. І. Марченко, М. М. Малько, М. І. Безменов – Х. : НТУ «ХПІ», 2017. – 20 с.

Укладачі: І. І. Марченко
М. М. Малько
М. І. Безменов

Рецензент Л. М. Любчик

Кафедра системного аналізу та інформаційно-аналітичних систем

ВСТУП

У процесі розв'язання великої кількості задач виникає потрібність у використанні та обробці різноманітної текстової інформації. Зазвичай процедура обробки інформації може містити такі етапи як введення, перетворення та виведення текстових даних.

Мета роботи – освоєння методики обробки текстової інформації.

1. ТЕОРЕТИЧНІ ОСНОВИ

1.1. Символьний тип та деякі його методи

В .NET Framework для подання символу Юникоду використовується тип `Char`. Значенням об'єкта `Char` є його 16-розрядний числовий код. У C# ключове слово `char` є псевдонімом для `Char`. Таким чином, `Char` і `char` є еквівалентні.

Символи можуть бути ініціалізовані константами. Символьні константи позначаються символом, розташованим між двома апострофами (наприклад, `'s'`). Для запису символу можуть також використовуватися так звані escape-послідовності. Для цього після зворотної косої указується символ `'u'` та шістнадцятковий код (шістнадцяткове число від `0000` до `FFFF` включно):

```
// два способи завдання символу 's'
char c0 = 's';
char c1 = '\u0073';
```

У мові C# визначена група керуючих символьних констант, що задаються комбінацією символу \ і деякого іншого символу. Ця комбінація укладається в апострофи при задаванні відповідної символьної константи.

Наявні такі керуючі символи:

`'\''` – апостроф;

`'\"'` – подвійна лапка;

`'\\'` – зворотна коса риска;

`'\0'` – нульовий символ;

`'\a'` – попередження;

`'\b'` – повернення на один символ (backspace);

`'\f'` – переведення сторінки;

`'\n'` – переведення рядка;

`'\r'` – повернення каретки;
`'\t'` – горизонтальна табуляція;
`'\U'` – escape-послідовність для сурогатних пар Юнікоду
`'\u'` – escape-послідовність для коду символу UTF-16
`'\v'` – вертикальна табуляція;
`'\x'` – escape-послідовність, що є аналогічною `'\u'`.

Тип `Char` надає ряд зручних методів для роботи з символами. Розглянемо деякі з них.

`Int32 Char.CompareTo(Char)` – порівнює даний екземпляр із заданим об'єктом `Char` і показує, чи розташований цей екземпляр перед, після або на тій же позиції в порядку сортування, що і заданий об'єкт `Char`. Відповіддю є від'ємне, 0 або додатне число:

```
char a = 'a';//'\u0061
char b = 'b';//'\u0063
Console.WriteLine(a.CompareTo(b));//61-63=-2,'a' меньше 'b'
Console.WriteLine(b.CompareTo(a));//63-61= 2,'b' більше 'a'
Console.WriteLine(a.CompareTo(a));//61-61=0,'a'дрівнює 'a'
```

`Boolean Char.IsDigit(Char)` – показує, чи відноситься символ Юнікоду до категорії десяткових цифр.

`Boolean Char.IsNumber(Char)` – визначає, чи є символ будь-якою числовою категорією Юнікоду. Окрім цифр, до категорії чисел включені дроби, індекси, римські цифри та інше:

```
char d0 = '1';
Console.WriteLine(Char.IsDigit(d0));   // true
Console.WriteLine(Char.IsNumber(d0));  // true

char d1 = '\u00BD'; //'½'
Console.WriteLine(Char.IsDigit(d1));   // false
Console.WriteLine(Char.IsNumber(d1));  // true
```

`Boolean Char.IsLetter(Char)` – визначає, чи відноситься символ Юнікоду до категорії літер.

`Boolean Char.IsLower(Char)` – показує, чи відноситься цей символ Юнікоду до категорії літер нижнього регістру.

`Boolean Char.IsUpper(Char)` – показує, чи відноситься цей символ Юнікоду до категорії літер верхнього регістру.

`Boolean Char.IsLetterOrDigit(Char)` – визначає, чи відноситься символ Юнікоду до категорії літер або десяткових цифр:

```
char e0 = 'e';
Console.WriteLine(Char.IsLetter(e0));        // true
Console.WriteLine(Char.IsDigit(e0));         // false
Console.WriteLine(Char.IsNumber(e0));        // false
Console.WriteLine(Char.IsLetterOrDigit(e0)); // true
Console.WriteLine(Char.IsLower(e0));         // true
Console.WriteLine(Char.IsUpper(e0));         // false

char e1 = 'E';
Console.WriteLine(Char.IsLower(e1));         // fasle
Console.WriteLine(Char.IsUpper(e1));         // true
```

`Boolean Char.`**`IsPunctuation`**`(Char)` – визначає, чи відноситься зазначений символ Юнікоду до категорії знаків пунктуації.

`Boolean Char.`**`IsSymbol`**`(Char)` – показує, чи відноситься цей символ Юнікоду до категорії допоміжних символів (символи знаків валют, математичні символи, технічні позначки тощо):

```
char f0 = ',';
Console.WriteLine(Char.IsPunctuation(f0));   //true
Console.WriteLine(Char.IsSymbol(f0));        //false


char f1 = '@';
Console.WriteLine(Char.IsPunctuation(f1));   //true
Console.WriteLine(Char.IsSymbol(f1));        //false


char f2 = '$';
Console.WriteLine(Char.IsPunctuation(f2));   //false
Console.WriteLine(Char.IsSymbol(f2));        //true
```

`Boolean Char.`**`IsControl`**`(Char)` – показує, чи відноситься даний символ Юнікоду до категорії керуючих символів.

`Boolean Char.`**`IsSeparator`**`(Char)` – визначає, чи відноситься вказаний символ Юнікоду до категорії знаків-роздільників.

`Boolean Char.`**`IsWhiteSpace`**`(Char)` – показує, чи відноситься цей символ Юнікоду до категорії пропусків:

```
char g0 = ' ';
Console.WriteLine(Char.IsControl(g0));          //false
Console.WriteLine(Char.IsSeparator(g0));        //true
Console.WriteLine(Char.IsWhiteSpace(g0));       //true

char g1 = '\n';
Console.WriteLine(Char.IsControl(g1));          //true
Console.WriteLine(Char.IsSeparator(g1));        //false
Console.WriteLine(Char.IsWhiteSpace(g1));       //true
```

`Char Char.ToLower(Char)` – перетворює значення символу Юникоду в його еквівалент в нижньому регістрі.

`Char Char.ToUpper(Char)` – перетворює значення символу Юникоду в еквівалентний символ верхнього регістру:

```
char h0 = 'h';
Console.WriteLine(Char.ToLower(h0));   //   h
Console.WriteLine(Char.ToUpper(h0));   //   H

char h1 = 'H';
Console.WriteLine(Char.ToLower(h1));   //   h
Console.WriteLine(Char.ToUpper(h1));   //   H

char h2 = '1';
Console.WriteLine(Char.ToLower(h2));   //   1
Console.WriteLine(Char.ToUpper(h2));   //   1
```

1.2. Тип String, деякі його властивості та методи

У мові C# рядок є об’єктом типу `String`. Всередині даного типу текстова інформація зберігається у вигляді впорядкованої колекції об’єктів `Char`, доступ до яких можливий тільки в режимі читання. У C# ключове слово `string` є псевдонімом для `String`. Таким чином, `String` і `string` можна застосовувати незалежно від використовуваної угоди про іменування.

Рядки можна оголошувати та ініціалізовувати різними способами, як наведено у наступному прикладі:

```
// оголошення без ініціалізації
string s1;
// ініціалізація значенням null.
string s2 = null;
// ініціалізація порожнього рядка літералом ""
string s3 = "";
// ініціалізація константою Empty.
string s4 = System.String.Empty;
// ініціалізація константним рядком.
string s5 = "Hello, pals!";
// створення рядка з масиву символів
char[] cc = { 'H', 'i', '!', '!', '!' };
string s6 = new string(cc); //  "Hi!!!"
```

Символ @ вказує, що при створенні рядка слід ігнорувати escape-послідовності та переноси рядка:

```
string path     = "c:\\home\\my";          // однакові
string eqvPath  = @"c:\home\my";           // рядки

string s1 = @"Hello,                       //
how r u doing?";                           // однакові
string s2 = @"Hello,\nhow r u doing?";     // рядки
```

Усі вбудовані типи даних C# надають метод **ToString**, що перетворює значення об'єктів у рядок:

```
double weight = 18;
string msg = "Улюблений хом'як важить" + weight.ToString() +
                                                " грамів";
Console.WriteLine(msg);
```

```
Улюблений хом'як важить 18 грамів                    console
```

Для роботи з різноманітною інформацією часто виникає необхідність у використанні форматних даних. Форматний рядок – це рядок, вміст якого можна визначити динамічно під час виконання. Для створення форматного рядка використовується статичний метод **Format** і задані у фігурних дужках заповнювачі (елементи форматування), які будуть замінені іншими значеннями під час виконання:

```
double weight = 18;
string s=string.Format("Улюблений хом'як важить {0} грамів",
                                                  weight);
Console.WriteLine(s);
```

```
Улюблений хом'як важить 18 грамів                    console
```

Метод `String.Format` повертає рядок, що був відформатований. Кожен елемент форматування складається з наступних компонентів `{index[,alignment][:formatString]}`.

Обов'язковий компонент `index`, що також називають описувачем параметра, – це ціле число, що визначає відповідний об'єкт зі списку параметрів (індексація елементів ведеться від нуля). Необов'язковий компонент `alignment` (вирівнювання) – це ціле число зі знаком, що служить для вказівки бажаної ширини поля форматування, а необов'язковий компонент `formatString` є рядком формату, який відповідає типу об'єкта, що форматується.

Одне з перевантажень методу `WriteLine` приймає як параметр форматний рядок. Тому можна просто застосувати рядковий літерал формату без явного виклику методу:

```
double weight = 18;
Console.WriteLine("Точніше, він важить {0:F2} грамів",
                                                    weight);
Console.WriteLine("Важить {0}, ні {1,-5},  ні
{2,5},точнісенько {0,0}!", weight, weight - 1, weight + 1);
```

```
Точніше, він важить 18.00 грамів                              console
Важить 18, ні 17   , ні    19,точнісенько 18!
```

У мові C# існує механізм створення інтерпольованих рядків. Інтерполяція по суті є більш лаконічним форматуванням. Наявний знак долара перед рядком вказує, що буде здійснюватися інтерполяція рядків.

Усередині рядка можна використати фігурні дужки `{...}`, у яких безпосередньо пишуться ті вирази, які необхідно вивести.

```
double weight = 18;
string s = $" Улюблений хом'як важить  {weight} грамів";
Console.WriteLine(s);
Console.WriteLine($"Точніше, він важить {weight:F2} грамів");
Console.WriteLine($"Важить {weight}, ні {weight - 1,-5}!");
```

```
Улюблений хом'як важить 18 грамів                             console
Точніше, він важить 18.00 грамів
Важить 18, ні 17   , ні    19!
```

Рядки можна об'єднувати, використовуючи оператор +:

```
string s1 = "Здоровенькі ";
string s2 = "були!";
string s = s1 + s2;
Console.WriteLine(s);
```

```
Здоровенькі були!                                             console
```

Такий саме результат можна отримати, якщо використати групу методів Concat.

`String String.Concat(String, String)` – зчіплює два зазначених екземпляра `String`.

`String String.Concat(String[])` – зчіплює елементи зазначеного масиву `String`:

```
string s1 = "Здоровенькі ";
string s2 = "були ";
string s = string.Concat(s1,s2);
Console.WriteLine(s);
s = string.Concat(new string[]{s1,s2,"шановні ", "друзі" });
Console.WriteLine(s);
Console.WriteLine(string.Concat(s1,", ", "шановні","друзі"));
```

```
Здоровенькі були                                    console
Здоровенькі були шановні друзі
Здоровенькі , шановні друзі
```

Рядкові об’єкти є незмінними, – їх не можна змінити після створення. Усі методи типу `String` і оператори C# повертають результати в новий рядковий об’єкт:

```
string s = "Здоро";
s += "венькі";//створюється новий строковий об'єкт для нової
            //послідовності знаків, і записується в змінну s
```

Як наведено в попередньому прикладі, оператор += створює новий рядок, який містить об’єднаний вміст. Цей новий об’єкт присвоюється змінній `s`, і початковий об’єкт, який був присвоєний `s`, звільняється для збору сміття, так як жодна змінна не посилається на нього.

Оскільки «змінення» рядка насправді є створенням нового об’єкту, використовувати посилання на рядки слід з обережністю. Якщо було створене посилання на рядок, а потім відбулося «змінення» початкового, посилання буде, як і раніше, вказувати на початковий об’єкт, а не на новий об’єкт, який був створений при змінюванні рядка:

```
string s1 = " Здоровенькі ";
string s2 = s1;
s1 += " були!";
System.Console.WriteLine(s1);
System.Console.WriteLine(s2);
```

```
Здоровенькі були!                                   console
Здоровенькі
```

Властивість `int String.Length` повертає кількість знаків у поточному об'єкті `String`:

```
string s = "хоп хоп хей";
Console.WriteLine(s.Length);
```

```
11                                                    console
```

Оператор `[]` використовується тільки для доступу до читання окремих знаків об'єкта `String`. Індекс – це положення об'єкта `Char` (а не знак Юнікоду) у `String`. Його значенням є невід'ємне ціле число, яке починається зі стартової позиції в рядку, що дорівнює нулю:

```
string s = "Hello, ";
char c = s[0];   //c : H
s[0] = "h";      //помилка компіляції
```

Група методів пошуку, такі як `IndexOf` і `LastIndexOf`, визначають індекс символу або підрядка в екземплярі рядка.

`Int32 String.IndexOf(String)` – повертає значення індексу з відліком від нуля першого входження значення вказаного рядка в даному екземплярі або значення `-1`, якщо підрядок не знайдено. Якщо значення підрядка дорівнює `String.Empty`, то повертається число `0`.

`Int32 String.IndexOf(String, Int32)` – аналогічний попередньому методу, але пошук починається із вказаної позиції символу:

```
string s = "хоп хоп хей";
Console.Write($"{s.IndexOf("хоп")} ");
Console.Write($"{s.IndexOf("хоп",0)} ");
Console.Write($"{s.IndexOf("хоп", 1)} ");
Console.Write(s.IndexOf("хоп", 7));
```

```
0 0 4 -1                                              console
```

`Int32 String.LastIndexOf(String)` – повертає позицію індексу з відліком від нуля останнього входження вказаного рядка в даному екземплярі або значення `-1`, якщо підрядок не знайдено. Якщо значення підрядка дорівнює `String.Empty`, то повертається `0`.

`Int32 String.LastIndexOf(String, Int32)` – аналогічний попередньому методу. Пошук починається із вказаної позиції символу і виконується у зворотному напрямку до початку рядка:

```
string s = "хоп хоп хей";
Console.Write($"{s.LastIndexOf("хоп")}");
Console.Write($"{s.LastIndexOf("хоп",0)} ");
Console.Write($"{s.LastIndexOf("хоп", 4)} ");
Console.Write(s.LastIndexOf("хоп", 7));
```

```
4 -1 0 4                                              console
```

`Int32 String.`**`IndexOfAny`**`(Char[])` – повертає індекс із відліком від нуля першого виявленого в даному екземплярі символу із зазначеного масиву символів Юнікоду або `-1`, якщо не знайдено жодного вказаного символу.

`Int32 String.`**`IndexOfAny`**`(Char[], Int32)` – аналогічний попередньому методу, але пошук починається з вказаної позиції символу:

```
string s = "хоп хоп хей";
char[] chrs = { 'х', 'п' };
Console.Write($"{s.IndexOfAny(chrs)} ");
Console.Write($"{s.IndexOfAny(chrs, 4)} ");
Console.Write($"{s.IndexOfAny(chrs, 7)} ");
Console.Write($"{s.IndexOfAny(chrs, 10));
```

```
0 4 8 -1                                                    console
```

Для роботи з рядками передбачений ряд методів.

`String String.`**`Substring`**`(Int32)` – видаляє підрядок з даного екземпляра. Підрядок починається в зазначеному положенні символів і досягає кінця рядка.

`String String.`**`Substring`**`(Int32, Int32)` – видаляє підрядок з даного екземпляра. Підрядок починається з вказаної позиції символу і має зазначену довжину:

```
string s = "Пане, рядок було вирізано";
Console.WriteLine(s.Substring(6));
Console.WriteLine(s.Substring(6, 5));
```

```
рядок було вирізано                                         console
рядок
```

`String String.`**`Remove`**`(Int32)` – повертає новий рядок, в якому були видалені всі символи, починаючи з вказаної позиції, і до його кінця.

`String String.`**`Remove`**`(Int32, Int32)` – повертає новий рядок, в якому було видалено вказане число символів, починаючи із зазначеної позиції:

```
string s = "Пане, рядок було видалено";
Console.WriteLine(s.Remove(6));
Console.WriteLine(s.Remove(6, 6));
```

```
Пане,                                                       console
Пане, було видалено
```

`String String.`**`Replace`**`(String, String)` – повертає новий рядок, в якому всі входження заданого рядка в поточному екземплярі замінені на інший заданий рядок:

```
string s = "Пане, рядок було змінено";
Console.WriteLine(s.Replace("Пане", "Добродію"));
Console.WriteLine(s.Replace("о", "О"));
```

```
Добродію, рядок було змінено                                    console
Пане, рядОк булО зміненО
```

`String String.`**`Insert`**`(Int32, String)` – повертає новий рядок, в якому зазначений рядок вставляється у вказану позицію індексу:

```
string s = "Пане, тут рядок";
Console.WriteLine(s.Insert(10, "було вставлено "));
```

```
Пане, тут було вставлено рядок                                  console
```

`String String.`**`Trim`**`()` – видаляє всі початкові і кінцеві символи-роздільники.

`String String.`**`Trim`**`(Char[])` – видаляє всі початкові і кінцеві входження набору символів, заданого у вигляді масиву.

`String String.`**`TrimStart`**`(Char[])` – видаляє всі початкові входження набору символів, заданого у вигляді масиву.

`String String.`**`TrimEnd`**`(Char[])` – видаляє всі кінцеві входження набору символів, заданого у вигляді масиву:

```
string s = "  !!  тут рядок  !!  ";
Console.WriteLine(s);
Console.WriteLine(s.Trim());
Console.WriteLine(s.TrimStart(new char[] { '!', ' ' }));
Console.WriteLine(s.TrimStart('!', ' '));
Console.WriteLine(s.TrimEnd('!', ' '));
Console.WriteLine(s.Trim('!',' '));
```

```
  !!  тут рядок  !!                                              console
!!  тут рядок  !!
тут рядок  !!
тут рядок  !!
  !!  тут рядок
тут рядок
```

`String String.`**`PadLeft`**`(Int32)` – повертає новий рядок, в якому символи вирівняні по правому краю шляхом додавання зліва символів-роздільників до вказаної загальної довжини.

`String String.`**`PadLeft`**`(Int32, Char)` – повертає новий рядок, в якому символи вирівняні по правому краю шляхом додавання зліва зазначеного символу Юникоду до вказаної загальної довжини.

`String String.`**`PadRight`**`(Int32)` – повертає новий рядок, в якому символи цього рядка вирівняні по лівому краю шляхом додавання справа символів-роздільників до вказаної загальної довжини.

`String String.PadRight(Int32, Char)` – повертає новий рядок, в якому символи цього рядка вирівняні по лівому краю шляхом додавання справа зазначеного символу Юникоду до вказаної загальної довжини:

```
string s = "тут рядок";
Console.WriteLine($"|{s.PadLeft(5)}|");
Console.WriteLine($"|{s.PadLeft(15)}|");
Console.WriteLine($"|{s.PadLeft(15,'!')}|");
Console.WriteLine($"|{s.PadRight(15)}|");
Console.WriteLine($"|{s.PadRight(15, '!')}|");
```

```
|тут рядок|                                                    console
|      тут рядок|
|!!!!!!тут рядок|
|тут рядок      |
|тут рядок!!!!!!|
```

`String[] String.Split(String[])` – розбиває рядок на підрядки, які розділені символами, що наведені в масиві.

`String[] String.Split(String[], StringSplitOptions)` – аналогічно попередньому. Можна вказати, чи включаються порожні підрядки до результуючого масиву:

```
string sd = "я  дуже  дуже  втомився";

string[] sa = sd.Split();
foreach (string s in sa)
    Console.Write($"{s}|");
Console.WriteLine();
sa=sd.Split(new string[]{" "},
                    StringSplitOptions.RemoveEmptyEntries);
foreach (string s in sa)
    Console.Write($"{s}|");
Console.WriteLine();

sa = sd.Split(new string[] { "дуже  "},
                    StringSplitOptions.None);
foreah (string s in sa)
     Console.Write($"{s}|");
```

```
я||дуже||дуже||втомився|                                    console
я|дуже|дуже|втомився|
я  ||втомився|
```

`String String.Join(String, String[])` – зв'язує всі елементи масиву рядків, розміщуючи між ними заданий роздільник:

```
string sd = "я  дуже  дуже  втомився";
string[] sa = sd.Split();
string sr = string.Join("*!*", sa);
Console.WriteLine(sr);
sa = sd.Split(new string[]{" "},
                    StringSplitOptions.RemoveEmptyEntries);
Console.WriteLine(string.Join(" ", sa));
```

```
я*!**!*дуже*!**!*дуже*!**!*втомився                    console
я дуже дуже втомився
```

Для порівняння рядків передбачено ряд методів.

`Boolean String.`**`Equals`**`(String)` – визначає, чи рівні значення поточного екземпляру і зазначеного об'єкта String.

`Boolean String.`**`Equals`**`(String, StringComparison)` – аналогічно попередньому. Параметр визначає мову і регіональні параметри, а також чутливість до регістру та правила сортування, що використовуються при порівнянні:

```
string msg1 = "Я втомився";
string msg2 = "Я ВТОМИВСЯ";
Console.Write(msg1.Equals(msg2));
Console.Write(msg1.Equals(msg2,StringComparison.Ordinal));
Console.Write(msg1.Equals(msg2,
                        StringComparison.OrdinalIgnoreCase))
```

```
False                                                   console
False
True
```

`Int32 String.`**`Compare`**`(String, String)` – порівнює два зазначених об'єкта String і повертає ціле число, яке показує їх відносне положення в порядку сортування (виражає лексичне відношення).

`Int32 String.`**`Compare`**`(String, String, Boolean)` – аналогічно попередньому. Порівняння йде з урахуванням або без урахування регістру:

```
string msg1 = "Я втомився";
string msg2 = "Я ВТОМИВСЯ";
Console.WriteLine(string.Compare(msg1,msg2));
Console.WriteLine(string.Compare(msg1, msg2,true));
```

```
-1                                                      console
0
```

`Int32 String.`**`Compare`**`(String, Int32, String, Int32, Int32)`– порівнює підрядки двох зазначених об'єктів `String` і повертає ціле число, яке показує їх відносне положення в порядку сортування. Другий і

четвертий параметр визначають позицію в рядках, з яких починається порівняння. Останній параметр вказує максимальну кількість порівнюваних знаків в підрядках.

`Int32 String.Compare(String, Int32, String, Int32, Int32, Boolean)` – аналогічно попередньому. Порівняння йде з урахуванням або без урахування регістру:

```
string msg1 = "Я втомився...";
string msg2 = "я дуже ВТОМИВСЯ!";
Console.WriteLine(string.Compare(msg1,2,msg2,7,8));
Console.WriteLine(string.Compare(msg1, 2, msg2, 7, 8, true));
```

```
-1                                                    console
0
```

`Boolean String.Contains(String)` – повертає значення, яке вказує, чи містить зазначений рядок значення підрядка, що передано як параметр.

`Boolean String.StartsWith(String)` – визначає, чи збігається початок даного екземпляра із вказаним рядком.

`Boolean String.StartsWith(String, StringComparison)` – визначає, чи збігається початок цього екземпляра рядка із заданим рядком при порівнянні з урахуванням заданого параметра порівняння.

`Boolean String.EndsWith(String)` – визначає, чи збігається кінець даного екземпляра рядка з вказаним рядком.

`Boolean String.EndsWith(String, StringComparison)` – визначає, чи збігається кінець екземпляра рядка із заданим рядком при порівнянні з урахуванням заданого параметра порівняння:

```
string msg = "Я дуже втомився";
Console.WriteLine(msg.Contains("дуже"));
Console.WriteLine(msg.StartsWith("дуже"));
Console.WriteLine(msg.StartsWith("я"));
Console.WriteLine(msg.StartsWith("я",
                          StringComparison.OrdinalIgnoreCase));
Console.WriteLine(msg.EndsWith("я"));
Console.WriteLine(msg.EndsWith("ся"));
```

```
True                                                  console
False
False
True
True
True
```

2. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

За час, відведений для виконання лабораторної роботи (2 академічні години), студент повинен:

1. Розробити алгоритм розв’язання задачі, запропонованої для програмування.
2. Здійснити програмну реалізацію розробленого алгоритму.
3. Здійснити налаштування програми, виправивши синтаксичні та логічні помилки.
4. Підібрати тестові дані для перевірки програми, включаючи винятковi випадки.
5. Відповісти на контрольні запитання.
6. Здати викладачу працездатну програму з демонстрацією її роботи на декількох варіантах вихідних даних.

3. ВАРІАНТИ ЗАДАЧ

1. Нехай надано рядок, що складається зі слів, відокремлених одне від одного одним і більше пропусками. Видалити в ньому всі слова, що складаються з трьох і менше символів.
2. Нехай надано рядок, що складається зі слів, відокремлених одне від одного одним і більше пропусками. Сформувати новий рядок, що складається зі слів початкового рядка. У результуючий рядок не включати ті слова, в яких є символи, що повторюються.
3. Нехай надано рядок, що складається зі слів, відокремлених одне від одного одним і більше пропусками. Сформувати новий рядок, що складається зі слів початкового рядка. У словах залишити символи, які зустрічаються тільки один раз (у тому порядку, в якому вони зустрічаються в слові).
4. Нехай надано рядок, що складається зі слів, відокремлених одне від одного одним і більше пропусками. Видалити в ньому всі символи, що повторюються поспіль
5. Нехай надано рядок, що складається зі слів, відокремлених одне від одного одним і більше пропусками. Так само дані слова S1 і S2. Сформувати новий рядок, що складається зі слів початкового рядка. Замінити в новому рядку кожне друге входження слова S1 на слово S2.
6. Нехай надано рядок, що складається з українських слів, відокремлених одне від одного одним і більше пропусками. Переписати в новий рядок слова, в яких голосні літери чергуються з приголосними.
7. Нехай надано рядок, що складається з українських слів, відокремлених одне від одного одним і більше пропусками. Сформувати новий рядок в якому усі літеральні назви цифр замінені відповідними арабськими цифрами.
8. Нехай надано рядок, що складається з українських слів, відокремлених одне від одного одним і більше пропусками. Сформувати новий рядок в якій усі арабські цифри замінені їх українськими назвами.
9. Нехай надано рядок, що складається зі слів, відокремлених одне від одного одним і більше пропусками. Визначити довжину найкоротшого із слів, що містяться в ньому.
10. Нехай надано рядок, що складається зі слів, відокремлених одне від одного одним і більше пропусками. Визначити кількість слів, що містяться в ньому і є паліндромами (однаково читаються зліва направо і справа наліво).

4. КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як описуються символьні змінні?
2. Як зберігаються текстові рядки в мові C#?
3. Як записуються символьні константи?
4. Як записуються керуючі символи та escape-послідовності?
5. Охарактеризуйте основні функції обробки символьних даних.
6. Як записуються рядкові константи?
7. Як обчислюється довжина рядка?
8. Перелічіть деякі методи копіювання рядків.
9. Які методи використовуються для зчеплення рядків?
10. Які методи використовуються для пошуку в рядках?
11. Як здійснюється порівняння рядків?

СПИСОК ЛІТЕРАТУРИ

1. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. / Дж. Рихтер ; пер. Е. Матвеев. – Санкт-Петербург: Питер, 2017. – 895 с.
2. Шилдт Г. Полный справочник по C# = C#: The Complete Reference / Г. Шилдт. – Москва: Вильямс, 2004. – 752 с.
3. Троелсен Э. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е издание = Pro C# 5.0 and the .NET 4.5 Framework, 6th edition / Э. Троелсен. – Москва: Вильямс, 2013. – 1312 с.
4. Фленов М. Е. Библия C#. 3-е изд. / М. Е. Фленов. – Санкт-Петербург: БХВ-Петербург, 2016. – 544 с.
5. Строки (Руководство по программированию на C#) / Microsoft. – URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/strings (дата звернення: 25.11.2017).
6. Класс String / Microsoft. – URL: https://msdn.microsoft.com/ru-ru/library/system.string(v=vs.110).aspx (дата звернення: 25.11.2017).

Навчальне видання

Методичні вказівки
до лабораторної роботи
«Робота з текстовою інформацією у .NET Framework»
з курсу «Кросплатформне програмування»
для студентів спеціальності 122 – Комп'ютерні науки

Укладачі: МАРЧЕНКО Ігор Іванович,
МАЛЬКО Максим Миколайович,
БЕЗМЕНОВ Микола Іванович.

Відповідальний за випуск О. С. Куценко
Роботу до видання рекомендував О. В. Горілий

За авторською редакцією

План 2017 р., поз. 191

Підписано до друку 30.09.2017 р. Формат 60×84 1/16. Папір офсетний.
RISO-друк. Гарнітура Таймс. Ум. друк. арк.0.9.
Наклад 50 прим. Зам. № 421-17. Ціна договірна

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 3657 від 27.12.2009 р.
61002, Харків, вул. Кирпичова, 2

Друкарня «ФОП Пісня О. В.»
Свідоцтво про державну реєстрацію ВО2 № 248750 від 13.09.2007 р.
61002, Харків, вул. Гіршмана, 16а, кв. 21, тел. (057) 764-20-28